09/025,133
Filed 2/17/98
GAU 2182
#20 (NE) Sub Spec 12-13-01

TITLE OF THE INVENTION

DATA TRANSMISSION APPARATUS, SYSTEM AND METHOD, AND IMAGE PROCESSING APPARATUS

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to a data transmission apparatus, system and method and an image processing apparatus, and more particularly, to a data transmission apparatus, system and method and an image processing apparatus used in a case where an image providing device such as a digital camera is directly connected to an image processing device such as a printer via a serial interface based on, e.g., the IEEE 1394 standards.

DESCRIPTION OF RELATED ART

Various types of systems which transfer data to a printer via a bus are known. For example, a known technique is to output data from a computer to the printer by using a defacto standard interface such as a

SCSI (Small Computer System Interface) or Centronics interface.

In other words, the printer is connected to a personal computer (PC) used as a host device via a parallel or serial interface such as a centronics or RS232C interface.

Further, digital devices as image providing devices such as a scanner, a digital still camera and a digital video camera, are also connected to the PC. Image data inputted by the respective digital devices is temporarily stored in a hard disk or the like on the PC, then processed by an application software program or the like on the PC and converted into print data for the printer, and transferred via the above interface to the printer.

In the above system, the PC has driver software programs, respectively, for controlling the digital devices and the printer. The image data outputted from the digital devices is held by these driver software programs in data format which can be easily handled and displayed on the PC. The stored data is converted into the print data by an image processing method in consideration of image characteristics of the input and output devices.

Today it is possible for a new interface such as an interface based on the IEEE 1394 standards (hereinafter referred to as "1394 serial bus") to directly connect an image providing device and a printer. In case of directly connecting the image providing device to the printer by the 1394 serial bus, an FCP (Function Control Protocol) operand may include print data. Further, in the 1394 serial bus, a register area for may be provided such that data transfer is performed by writing data into the register area.

Further, as the 1394 serial bus has a plurality of data-transfer control procedures, data transfer can be performed in methods appropriate to the respective devices.

Further, the 1394 serial bus has an isochronous transfer mode and an asynchronous transfer mode. Time-restricted data, e.g., real-time data, is transferred by isochronous transfer, while simple data transfer is performed by asynchronous transfer.

As described above, the image data outputted from the image providing device is converted into print data by the PC and print-outputted by the printer, accordingly, even if the image providing device and the printer are directly connected, printing cannot be

performed without the PC. A video printer which directly print-outputs image data outputted from a digital video camera is known, however, even in case of using this printer, connection is made only between specific devices. There is no video printer which is directly connected to a number of image providing devices for general printing purposes. That is, it is impossible to directly send image data from the image providing device to a printer for printing, by utilizing a function to directly connect devices, which is characteristic of the 1394 serial bus or the like.

In the above method which directly connects the image providing device to the printer with the 1394 serial bus and includes print data into an FCP operand, the control commands cannot be separated from the print data. Further, in this method, the transfer efficiency is low since a response is always required with respect to a command. The above method providing a register area for data transfer requires processing to determine whether or not data can be written into the register area at every data transfer. Accordingly, the overhead of the determination processing is great, which degrades the transfer efficiency.

Further, to ensure connection with various types

of devices, a plurality of data transfer methods appropriate to the respective devices are required.

The data transfer methods are briefly classified into a PUSH method in which a host device writes data into a target device and a PULL method in which a target device reads data from a host device. One of these transfer methods is determined in accordance with resources of both host and target devices. However, in case of directly connecting with various types of devices, it is impossible to determine which method, the PUSH method or the PULL method, as the data transfer method.

Further, as the basic transfer method of in the synchronous mode is different from the asynchronous mode, it is difficult to change the synchronous and asynchronous modes in the same transfer procedure.

SUMMARY OF THE INVENTION

The present invention has been made to solve the respective or all of the above problems, and has its object to provide a data transmission apparatus, system and method and an image processing apparatus, appropriate for directly connecting a host device with a target device by using a serial interface based on the

1394 serial bus or the like and processing image data, directly sent from the host device to the target device, by the target device.

Further, another object of the present invention is to provide a data transmission apparatus, system and method and an image processing apparatus which separate control commands from data and attain high transfer efficiency.

Further, another object of the present invention is to provide a data transmission apparatus, system and method and image processing apparatus which set a data transfer method appropriate to a connected device.

Further, another object of the present invention is to provide a data transmission apparatus, system and method and image processing apparatus which perform data transfer by a PULL method.

According to the present invention, the foregoing objects are attained by providing a data transmission method for host and target devices which are connected by a serial bus, the method comprising the steps of: performing bi-directional communication between the host and target devices; and setting a data transfer method to be performed from a plurality of data transfer methods including a PULL model in which the target

device reads data from the host device, based on the bi-directional communication.

Further, the foregoing objects are attained by providing an image processing apparatus comprising: communication means for performing communication with a target device by the data transfer method in the above data transmission method; and transmission means for transmitting image data to the target device via the communication means.

Further, the foregoing objects are attained by providing a data transmission system for transferring data through a serial bus, comprising: communication means for performing bi-directional communication between host and target devices; and setting means for setting a data transfer method to be set from a plurality of data transfer methods including a PULL model, based on the bi-directional communication.

Further, another object of the present invention is to provide a data transmission apparatus, system and method and an image processing apparatus which perform synchronous transfer and asynchronous transfer in the same transfer procedure.

According to the present invention, the foregoing object is attained by providing a data transmission

method of host and target devices which are connected by a serial bus, the method comprising the steps of: transferring data from the host device to the target device, by isochronous transfer or asynchronous transfer; and transferring a procedure signal for transfer of the data to the host and target devices by common asynchronous transfer.

Further, the foregoing object is attained by providing a data transmission apparatus connected to a serial bus, comprising: transfer means for transferring a procedure signal for data transfer by asynchronous transfer common to a target device; and transmission means for transmitting data to be transmitted to the target device by isochronous transmission or asynchronous transmission.

Further, the foregoing object is attained by providing a data transmission system for transferring data through a serial bus, comprising: first transfer means for transferring a procedure signal for data transfer by common asynchronous transfer to host and target devices; and second transfer means for performing data transfer between the host and target devices by isochronous transfer or the asynchronous transfer.

Other features and advantages of the present

invention will be apparent from the following description taken in conjunction with the accompanying drawings, in which like reference characters designate the same name or similar parts throughout the figures thereof.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of the specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the invention.

Fig. 1A is an explanatory view showing a general construction of a system to which the present invention is applied;

Fig. 1B is a block diagram showing an example of a network system constructed with an IEEE 1394 serial interface;

Fig. 2 is a block diagram showing the construction of the IEEE 1394 serial interface;

Fig. 3 is an explanatory view showing address space of the IEEE 1394 serial interface;

Fig. 4 is a cross-sectional view showing a cable

for the IEEE 1394 serial interface;

Fig. 5 is a timing chart explaining a Data/Strobe Link method;

Figs. 6 to 8 are flowcharts showing a procedure of network construction in the IEEE 1394 serial interface;

Fig. 9 is a block diagram showing an example of the network;

Figs. 10A and 10B are block diagrams explaining bus arbitration;

Fig. 11 is a flowchart showing a procedure of the bus arbitration;

Fig. 12 is a timing chart showing transitional statuses in asynchronous data transfer;

Fig. 13 is a diagram showing a packet format for the asynchronous transfer;

Fig. 14 is a timing chart showing transitional statuses in isochronous data transfer;

Fig. 15A is an example of a packet format for the isochronous transfer;

Fig. 15B is a table showing the details of packet format fields for the isochronous transfer in a 1394 serial bus;

Fig. 16 is a timing chart showing transitional statuses in data transfer on the bus when the

isochronous transfer and asynchronous transfer are mixedly performed;

Fig. 17 is a schematic view showing the IEEE 1394 serial interface in comparison with an OSI model;

Fig. 18 is an explanatory view showing the basic operation of a LOGIN protocol;

Fig. 19 is an explanatory view showing connection status in the IEEE 1394 serial interface;

Fig. 20 is a timing chart showing the flow of log-in operation;

Fig. 21 is a schematic view showing a CSR preparing respective devices;

Fig. 22 is a flowchart showing LOGIN processing in a host device;

Fig. 23 is a flowchart showing LOGIN processing in a target device;

Fig. 24 is an explanatory view showing a case in consideration of a device without the LOGIN protocol;

Fig. 25 is a schematic view showing the IEEE 1394 serial interface in comparison with an OSI model, in the second embodiment;

Fig. 26A is a table showing functions of a CSR architecture of the 1394 serial bus;

Fig. 26B is a table showing registers for the 1394

serial bus;

Fig. 26C is a table showing registers for node resources of the 1394 serial bus;

Fig. 26D is an example of a minimum format of a configuration ROM of the 1394 serial bus;

Fig. 26E is an example of a general format of the configuration ROM of the 1394 serial bus;

Fig. 27A is a timing chart showing a request-response protocol with read, write and lock commands, based on the CSR architecture in a transaction layer;

Fig. 27B is a timing chart showing services in a link layer;

Fig. 28A is a timing chart showing an operation example of a split transaction;

Fig. 28B is an explanatory view showing transitional statuses of transfer by the split transaction;

Fig. 29 is an explanatory view showing an AV/C transaction in the 1394 serial bus;

Fig. 30 is an example of the packet format for the asynchronous transfer including an FCP packet frame;

Fig. 31 is an example of the structure of an AV/C command frame;

Fig. 32 is an example of the structure of an AV/C

response frame;

Fig. 33 is a schematic view showing a register map;

Fig. 34 is an explanatory view showing the flow of frames from an image providing device to the printer;

Fig. 35 is an explanatory view showing the construction of a format register;

Fig. 36 is an explanatory view showing the detailed construction of a status register of a common register group;

Fig. 37 is an explanatory view showing the details of information held in a register GLOBAL of a common register group;

Fig. 38 is an explanatory view showing the details of information held in a register LOCAL of the common register group;

Fig. 39 is an explanatory view showing the details of information held in a register format[1];

Fig. 40 is an explanatory view showing the details of information held in a register format[2];

Fig. 41 is a table showing commands and responses to the commands;

Fig. 42 is an example of an image data formats supported by the DPP protocol;

Fig. 43 is a timing chart showing the flow of format setting processing;

Fig. 44 is a timing chart showing the flow of data-transfer method setting processing;

Fig. 45 is an explanatory view showing a register map of registers necessary for transfer method 1 and transfer method 2, in address space of the 1394 serial bus;

Fig. 46 is an example of a data packet frame;

Fig. 47 is an example of the structure of a data header;

Fig. 48 is an explanatory view showing data packet frame processing in the printer in block transfer;

Fig. 49 is a timing chart showing commands and responses FreeBlock in the transfer method 1;

Fig. 50 is a timing chart showing the flow of data transfer in the transfer method 1;

Fig. 51 is a timing chart showing the flow of data transfer in the transfer method 2;

Fig. 52 is a timing chart showing the commands and responses FreeBlock in the transfer method 1, in detail;

Fig. 53 is a timing chart showing a commands and responses WriteBlock in the transfer method 1 and the transfer method 2;

Fig. 54 is a timing chart showing the commands and responses WriteBlock in the transfer method 1 and the transfer method 2, in detail;

Fig. 55 is a timing chart showing the flow of WriteBlock error processing upon occurrence of bus reset;

Fig. 56 is an explanatory view showing the construction of command registers, response registers and data registers of the image providing device and the printer, in a PUSH Large Buffer Model;

Fig. 57 is a timing chart showing the flow of operation in a PUSH buffer model between the image providing device and the printer;

Fig. 58 is an example of a data packet in a data frame;

Fig. 59 is an explanatory view of the relation between the data register and the buffer of the printer;

Fig. 60 is an explanatory view showing the construction of the command registers, the response registers and the data registers of the image providing device and the printer, in a PULL buffer model;

Fig. 61 is a timing chart showing the flow of operation in the PULL buffer model between the image providing device and the printer;

Fig. 62 is an explanatory view showing the relation between the data register and the buffer of the image providing device;

Fig. 63 is an explanatory view showing memory allocation for command registers and response registers for flow control; and

Fig. 64 is a timing chart showing the flow of print processing.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Hereinbelow, a data transfer method according to an embodiment of the present invention will now be described in detail in accordance with the accompanying drawings.

Fig 1A shows an example of general construction of a system to which the present invention is applied, where a PC 103, a printer 102 and a digital video camera (DVC) 101 are connected via a 1394 serial bus. Then the outline of the 1394 serial bus will be described below.

[Outline of 1394 Serial Bus]

With the appearance of general digital video cam recorder (VCR) and digital video disk (DVD) player, there is a need for transferring realtime and large

amount data such as video data and audio data (hereinafter referred to as "AV data"). To transfer AV data in realtime to a personal computer (PC) or other digital devices, an interface capable of high-speed data transfer is required. The 1394 serial bus has been developed from the above point.

Fig. 1B shows an example of a network system constructed with a 1394 serial bus. This system comprises devices A to H, and the devices A and B, the devices A and C, the devices B and D, the devices D and E, the devices C and F, the devices C and G, and the device C and H are respectively connected by a twisted pair cable for the 1394 serial bus. These devices A to H may be computers such as a personal computer, or most computer-peripheral devices such as a digital VCR, a DVD player, a digital still camera, a storage device using a storage medium such as a hard disk or an optical disk, a monitor such as a CRT or an LDC, a tuner, an image scanner, a film scanner, a printer, a MODEM, and a terminal adapter (TA).

Note that the printing method of the printer may be any method, e.g., a laser-beam printing, an electrophotographic method using an LED, an ink-jet method, a thermal-transfer method of ink melting or ink

sublimation type and a thermo-sensitive printing method.

The connection between the devices may be made by mixedly using a daisy chain method and a node branching method, thus realizing high-freedom of connecting.

The respective devices have an ID, and they construct a network by identifying each ID within a range connected by the 1394 serial bus. For example, the devices respectively take a relaying role only by daisy-chain connecting the devices with cables for the 1394 serial bus, thus constructing a network.

As the 1394 serial bus corresponds to Plug and Play function, it automatically recognizes a device connected to the cable, thus recognizes connection status. In the system as shown in Fig. 1B, when a device is removed from the network, or a new device is added to the network, the bus is automatically reset (i.e., the current network constructing information is reset), and a new network is constructed. This function enables realtime setting and recognition of network construction.

The 1394 serial bus has a data transfer speed defined as 100/200/400 Mbps. A device having a high transfer speed supports a lower transfer speed, thus maintaining compatibility. As data transfer modes, an asynchronous transfer mode (ATM) for transferring

asynchronous data such as a control signal, an isochronous transfer mode for transferring isochronous data such as realtime AV data are available. In data transfer, within each cycle (generally 125 µs/cycle), a cycle start packet (CSP) indicating the start of cycle is transferred, and then asynchronous and isochronous data are mixedly transferred such that the isochronous data transfer is transferred prior to the asynchronous data.

Fig. 2 shows the construction of the 1394 serial bus, as a layer structure. As shown in Fig. 2, a connector port 810 is connected to a connector at the end of a cable 813 for the 1394 serial bus. A physical layer 811 and a link layer 812 in a hardware unit 800 are positioned as upper layers with respect to the connector port 810. The hardware unit 800 comprises interface chips. The physical layer 811 performs coding, connection-related control and the like, and the link layer 812, packet transfer, cycle-time control and the like.

In a firmware unit 801, a transaction layer 814 manages data to be transferred (transaction data), and outputs commands Read, Write and Lock. A management layer 815 in the firmware unit 801 manages connection

statuses and ID's of the respective devices connected to the 1394 serial bus, thus manages the network construction. The above hardware and firmware units substantially constructs the 1394 serial bus.

In a software unit 802, an application layer 816 differs in software used by the system, and the data transfer protocol indicating how to transfer data on the interface is defined by a protocol such as a printer protocol or an AVC protocol.

Fig. 3 shows address space of the 1394 serial bus. All the devices (nodes) connected to the 1394 serial bus have a unique 64 bit address. The 64 bit address is stored in a memory of the devices. Data communication with a designated destination device can be performed by always recognizing the node addresses of the transmitting- and receiving-side nodes.

Addressing of the 1394 serial bus is made based on the IEEE 1212 standards, such that first 10 bits are allocated for designating a bus number, then next 6 bits are allocated for designating an node ID.

48-bit address used in the respective devices are divided into 20 bits and 28 bits, and utilized in the unit of 256 Mbytes. In the initial 20-bit address space, "0" to "0xFFFFD" is called a memory space; "0xFFFFE", a

private space; "0xFFFFF", a register space. The private space is an address freely used in the device. The register space, holding information common to the devices connected with the bus, is used for communication among the respective devices.

In the register space, the initial 512 bytes are assigned to a register core (CSR core) as a core of a Command/Status Register (CSR) architecture; the next 512 bytes, to a register of the serial bus; the next 1024 bytes, to a configuration ROM; and the remaining bytes, to a register unique to the device in a unit space.

Generally, for the sake of simplification of bus system design for different node types, it is preferable that only the initial 2048 bytes are used for the nodes, and as a result, total 4096 bytes are used including the initial 2048 bytes for the CSR core, the register of the serial bus, the configuration ROM and the unit space.

The 1394 serial bus has the construction as described above. Next, the features of the 1394 serial bus will be described in more detail.

[Electrical Specification of 1394 Serial Bus]

Fig. 4 shows a cross-section of the cable of the 1394 serial bus. The 1394 serial cable comprises two

sets of twisted pair signal lines and two power-source lines. This construction enables power supply to a device which lacks a power source, or a device where a voltage is degraded due to a failure or the like. The direct-current voltage supplied by the power-source lines is 8 to 40V; the current is maximum 1.5 A. Note that in the standards for so-called DV cable, four lines except the power-source line construct the cable.

[DS-Link]

Fig. 5 is a timing chart explaining a DS-Link (Data/Strobe-Link) method as a data transfer method.

The DS-Link method, appropriate for high-speed serial data communication, requires two sets of two signal lines. That is, one of the two sets of twisted-pair signal lines is used for sending a data signal, and the other one set of twisted-pair signal lines is used for sending a strobe signal. On the receiving side, an EXCLUSIVE-OR between the data signal and the strobe signal is obtained so as to generate a clock signal. In the DS-Link transfer, it is unnecessary to mix a clock signal into a data signal, therefore, transfer efficiency is higher than that in other serial-data transfer methods. Further, as a clock signal is

generated from the data signal and the strobe signal, a phase locked loop (PLL) circuit can be omitted, which attains downsizing of the scale of a controller LSI. Further, in the DS-Link transfer, it is unnecessary to send information indicative of idle status when there is no data to be transferred, therefore, a transceiver of each device can be set in a sleep status, which reduces electric consumption.

[Bus-Reset Sequence]

The respective devices (nodes) connected to the 1394 serial bus are provided with a node ID, and are recognized as nodes constructing the network. For example, when increase/decrease of the number of nodes due to connection/disconnection or power ON/OFF status of network devices, i.e., network construction changes and it is necessary to recognize a new network construction, the respective nodes detect the change of network construction, send a bus-reset signal onto the bus, and enter a mode for recognizing the new network construction. The detection of change of network construction is made by detecting change of bias voltage at the connector port 810.

When the bus-reset signal is sent from one node,

the physical layer 811 of the respective nodes receives the bus-reset signal, and at the same time, notifies the link layer 812 of the occurrence of bus reset, and forwards the bus-reset signal to the other nodes. When all the nodes have received the bus-reset signal, a bus-reset sequence is started. Note that the bus-reset sequence is started when the cable is attached/detached, or the hardware unit 800 has detected network abnormity or the like. Further, the bus-reset sequence is also started by a direct instruction to the physical layer 811 such as host control by a protocol. As the bus-reset sequence is started, data transfer is suspended during the bus reset, and after the bus reset, the data transfer is restarted in the new network construction.

[Node-ID Determination Sequence]

After the bus reset, the respective nodes start to obtain a node ID so as to construct a new network construction. A general sequence from the bus reset to node-ID determination will be described with reference to the flowcharts of Figs. 6 to 8.

Fig. 6 is a flowchart showing a sequence from occurrence of bus-reset signal to node-ID determination and data transfer. At step S101, the respective nodes

always monitor occurrence of bus-reset signal. When the bus-reset signal has occurred, the process proceeds to step S102, at which to obtain a new network construction in a state where the network construction has been reset, parent-child relation is declared between nodes connected to each other. Step S102 is repeated until it is determined at step S103 that the parent-child relation has been determined among all the nodes.

As the parent-child relation has been determined, the process proceeds to step S104, at which one "root (node)" is determined. At step S105, node-ID setting is performed so as to provide an ID to the respective nodes. The node-ID setting is made in a predetermined order of the nodes. Step S105 is repeated until it is determined at step S106 that the ID's have been given to all the nodes.

As the node-ID setting has been completed, since the new network construction has been recognized by all the nodes, data transfer among the nodes is possible. At step S107, data transfer is started, and the process returns to step S101, at which occurrence of bus-reset signal is monitored again.

Fig. 7 is a flowchart showing the sequence from the monitoring of bus-reset signal (S101) to the root

determination (S104) in detail. Fig. 8 is a flowchart showing the node-ID setting (S105 and S106) in detail.

In Fig. 7, at step S201, the occurrence of bus-reset signal is monitored, and as the bus-reset signal has occurred, the network construction is reset. Next, at step S202, as a first step for re-recognizing the reset network construction, the respective devices reset its flag FL with data indicative of "leaf (node)". At step S203, the respective devices examine the number of ports, i.e., the number of other nodes connected to them. At step S204, based on the result of examination at step S203, the devices examine the number of undefined (i.e., parent-child relation has not been determined) ports. The number of undefined ports is equal to that of the ports immediately after the bus reset, however, with the progress of determination of parent-child relation, the number of undefined ports detected at step S204 decreases.

Only actual leaf(ves) can declare parent-child relation immediately after the bus reset. Whether or not the node is a leaf is detected from the number of ports examined at step S203; i.e., if the number of ports is "1", the node is a leaf. The leaf declares that "this node is a child, and the connected node is a parent" at

step S205, then terminates the operation.

On the other hand, a node that detected at step S203 that the number of ports is "two or more" is a "branch". Immediately after the bus reset, as "undefined ports > 1" holds, the process proceeds to step S206, at which the flag FL is set with data indicative of "branch", then declaration of parent-child relation from another node is waited at step S207. When the parent-child relation is declared from another node, the process returns to step S204 at which the branch examines the number of undefined ports. If the number of undefined ports is "1", the branch can declare at step S205 that "this node is a child, and the connected node is a parent" to the node connected to the remaining port. If the number of undefined ports is still "two or more", the branch waits for declaration of parent-child relation from another node at step S207.

When any one of the branches (or exceptionally leaf(ves) which delayed declaring a child) detects that the number of undefined ports is "0", the parent-child declaration of the overall network has been completed. The only one node that has "0" undefined port, i.e., the parent of all the nodes, sets the flag FL with data indicative of a "root" at step S208. Then at step S209,

the node is recognized as a root.

In this manner, the procedure from the bus reset to the parent-child declaration among all the nodes in the network ends.

Next, a procedure of providing each node with an ID will be described. First, the ID setting is performed at the leaves. Then, ID's are set in numerical order (from node number: 0) from leaves → branches → root.

In Fig. 8, at step S301, the process splits in accordance with node type, i.e., leaf, branch or root, based on the data set at the flags FL.

In case of leaf, at step S302, the number of leaves (natural number) in the network is set to a variable N. At step S303, the respective leaves request a node number to the root. If a plurality of requests have been made, the root performs arbitration at step S304, and provides a node number to one node at step S305, while notifies the other nodes of the result of acquisition of node-number indicating that the node number has been failed.

A leaf that has not obtained a node number (NO at step S306) repeats the request for node number at step S303. On the other hand, a leaf that has obtained a node number notifies all the nodes of the obtained node

number by broadcasting ID information including the node number. As the broadcasting of the ID information has been completed, the variable N indicative of the number of leaves is decremented at step S308. Then, from the determination at step S309, the procedure from step S303 to step S308 is repeated until the variable N becomes "0" in the determination at step S309. When ID information on all the leaves have been broadcasted, the process proceeds to step S310, for setting ID's of branches.

The ID setting for branches is performed substantially similar to the ID setting for the leaves. First, at step S310, the number of branches (natural number) is set to a variable M. At step S311, the respective branches request the root for a node number. In response to the requests, the root performs arbitration at step S312, and provides a node number, subsequent to the last leaf node number, to a branch at step S313, while notifies the other branches of the result of acquisition of node-number indicating that the node number has been failed.

A branch that has not obtained a node number (NO at step S314) repeats the request for node number at step S315. On the other hand, a branch that has obtained

a node number notifies all the nodes of the obtained node number by broadcasting ID information including the node number. As the broadcasting of the ID information has been completed, the variable M indicative of the number of branches is decremented at step S316. Then, from the determination at step S317, the procedure from step S311 to step S316 is repeated until the variable M becomes "0" in the determination at step S317. When ID information on all the leaves have been broadcasted, the process proceeds to step S318, for setting the ID of the root.

At this time, it is only the root that has not obtained a node ID. At step S318, the root obtains the smallest number that has not been provided to any other node as the node ID of the root, and at step S319, broadcasts ID information on the root.

As described above, the procedure until the node ID's for all the nodes have been set ends. Next, the sequence of node ID determination will be described with reference to the network example shown in Fig. 9.

In the network in Fig. 9, a node B as a root is directly connected to its lower nodes A and C; the node C is directly connected to its lower node D; and the node D is directly connected to its lower nodes E and F.

The procedure of determining this hierarchical structure, the root node and the node ID's will be described below.

After the bus reset has occurred, to recognize connection statuses of the respective nodes, parent-child relation is declared between ports of directly connected nodes. "parent" means a node at an upper level and "child" means a node at a lower level in the hierarchical structure. In Fig. 9, the node that first declared parent-child relation after the bus reset is the node A. As described above, nodes (leaves) where only one port is connected can start declaration of parent-child relation. That is, if the number of ports is "1", it is recognized that the node is the end of the network tree, i.e., a leaf. The declaration of parent-child relation is started from the leaf which has first taken action among these leaves. Thus, a port of the leave node is set as a "child", while the port of another node connected to the leave node is set as a "parent". In this manner, "child-parent" relation is sequentially set between the nodes A and B, between the nodes E and D, and between the nodes F and D.

Further, among upper nodes having a plurality of ports, i.e., branches, parent-child relation is sequentially declared with respect to upper node(s),

from the node that first received declaration of parent-child relation from the leaf. In Fig. 9, first parent-child relation is determined between the nodes D and E and between the nodes D and F. Then the node D declares parent-child relation with respect to the node C, and as a result, a relation "child-parent" is set between the nodes D and C. The node C, that has received the declaration of parent-child relation from the node D, declares parent-child relation with respect to the node B connected to the other port, thus "child-parent" relation is set between the nodes C and B.

In this manner, the hierarchical structure as shown in Fig. 9 is constructed. The node B, that has finally become the parent at all the ports, is determined as a root. Note that a network has only one root. In a case where the node B that has received declaration of parent-child relation from the node A immediately declares parent-child relation with respect to another node, the other node, e.g., the node C, may be the root node. That is, any node may be a root depending upon timing of transmitting declaration of parent-child relation, and further, even in a network maintaining the same construction, a particular node is not always become a root.

As the root has been determined, the sequence of determining the respective node ID's is started. Each node has a broadcast function to notify its ID information to all the other nodes. ID information includes a node number, information on a connected position, the number of ports, the number of ports connected to other nodes, information on parent-child relation on the respective ports and the like.

As described above, the assignment of node numbers is started from the leaves. In numerical order, node number = 0, 1, 2,.... is assigned. Then, by the broadcasting of ID information, it is recognized that the node number has been assigned.

As all the leaves have obtained a node number, node numbers are assigned to the branches. Similar to the assignment of node numbers to the leaves, ID information is broadcasted from the branch that received a node number, and finally, the root broadcasts its ID information. Accordingly, the root always has the greatest node number.

Thus, as the ID setting of the overall hierarchical structure has been completed and the network has been constructed, then the bus initialization is completed.

[Control Information for Node Management]

The CSR core as shown in Fig. 3 exists on the register as a basic function of the CSR architecture for node management. Fig. 26A shows the positions and functions of the registers. In Fig. 26A, the offset is a relative position from "0xFFFFF0000000.

In the CSR architecture, the register for the serial bus is arranged from "0xFFFFF0000200". Fig. 26B shows the positions and functions of the registers.

Further, information on node resources of the serial bus is arranged from "0xFFFFF0000800". Fig. 26C shows the positions and functions of the registers.

The CSR architecture has a configuration ROM for representing functions of the respective nodes. The configuration ROM has a minimum format and a general format, arranged from "0xFFFFF0000400". As shown in Fig. 26D, the minimum format configuration ROM merely shows a vendor ID which is a unique numerical value represented by 24 bits.

As shown in Fig. 26E, the general format configuration ROM has information on a node. In this case, the vendor ID in this format is included in a "root_directory". Further, "bus_info_block" and

"root&unit_leaves" include unique device number including the vendor ID, represented by 64 bits. The device number is used after network reconstruction by bus reset operation, to continue recognition of the node.

[Serial Bus Management]

As shown in Fig. 2, the protocol of the 1394 serial bus comprises a physical layer 811, a link layer 812 and a transaction layer 814. This provides, as the serial bus management, a basic function for node management and bus resource management, based on the CSR architecture.

Only one node which performs bus management (hereinafter referred to as "bus management node") exists on the same bus, and provides the other nodes on the serial bus with management function which includes cycle master control, performance optimization, power source management, transmission speed management, construction management and the like.

The bus management function briefly divides into a bus manager, an isochronous resource manager and a node control function. The node control is a management function which enables communication among the nodes in the physical layer 811, the link layer 812, the link

layer 812, the transaction layer 814 and an application program by the CSR. The isochronous resource manager, which is a management function necessary for isochronous-type data transfer on the serial bus, manages assignment of transfer bandwidth and channel number to isochronous data. For this management, after bus initialization, the bus management node is dynamically selected from nodes having the isochronous resource manager function.

Further, in a construction without a bus management node on the bus, a node having the isochronous resource manager function performs a part of the bus management such as the power source management and cycle master control. Further, the bus management is a management function as service to provide a bus control interface to an application program. The control interface uses a serial-bus control request (SB_CONTROL.request), a serial-bus event control confirmation (SB_CONTROL.confirmation) and a serial-bus event indication (SB_EVENT.indication).

The serial-bus control request is used when an application program requires the bus management node to perform bus reset, bus initialization, representation of bus-status information, and the like. The serial-bus

event control confirmation is the result of the serial-bus control request, and is notified from the bus management node to the application for confirmation. The serial-bus event control confirmation is made as notification of an asynchronously-caused event from the bus management node to the application.

[Data Transfer Protocol]

The data transfer by using the 1394 serial bus simultaneously sends isochronous data (isochronous packet) which must be periodically transmitted, and asynchronous data (asynchronous packet) which can be transmitted/received at arbitrary timing, further, ensures real-time transmission of isochronous data. In the data transfer, a bus use right is requested prior to transfer, and bus arbitration is performed to obtain bus use permission.

In the asynchronous transfer, a transmitting node ID and a receiving node ID are sent with transfer data as packet data. The receiving node confirms the receiving node ID, i.e., its node ID, receives the packet, and returns an acknowledge signal to the transmitting node. Thus, one transaction is completed.

In the isochronous transfer, a transmitting node

requires an isochronous channel with a transmission speed, and a channel ID is sent with transfer data as packet data. A receiving node confirms a desired channel ID and receives the data packet. The necessary channel number and transmission speed are determined by the application layer 816.

These transfer protocols are defined by the physical layer 811, the link layer 812 and the transaction layer 813. The physical layer 811 performs physical and electrical interface with the bus, automatic recognition of node connection, bus arbitration for a bus use right among nodes, and the like. The link layer 812 performs addressing, data checking, packet transmission/reception and cycle control for isochronous transfer. The transaction layer 814 performs processing relating to asynchronous data. Hereinbelow, the processings in the respective layers will be described.

[Physical Layer]

The bus arbitration in the physical layer 811 will be described.

The 1394 serial bus always performs arbitration of bus use right prior to data transfer. The devices

connected to the 1394 serial bus respectively relay a signal transferred on the network, thus constructing a logical bus-type network transmitting the signal to all the devices within the network. This necessitates bus arbitration to avoid packet conflict. As a result of bus arbitration, one node can transfer data during a certain period.

Figs. 10A and 10B are block diagrams explaining the bus arbitration. Fig. 10A shows operation to request a bus use right; and Fig. 10B, operation to allow to use the bus.

When the bus arbitration is started, a single or plurality of nodes respectively request a bus use right to use the bus to its parent node. In Fig. 10A, the nodes C and F request a bus use right. The parent node (node A in Fig. 10A) that has received the request relays the request by further requesting a bus use right to its parent node. The request is forwarded to a root (node B in Fig. 10A) that finally performs arbitration.

The root that has received the request for bus use right determines a node to be provided with the bus use right. This arbitration can be performed only by the root. The node that dominated in the arbitration is provided with the bus use right. Fig. 10B shows that the

node C has obtained the bus use right and the request from the node F has been rejected.

The root sends a DP (data prefix) packet to nodes lost in the bus arbitration so as to notify that their requests have been rejected. The requests from those nodes are held by the next bus arbitration.

Thus, the node that obtained the bus use permission starts data transfer.

The sequence of the bus arbitration will be described with reference to the flowchart of Fig. 11.

To start data transfer by a node, the bus must be in idle status. To confirm that data transfer has been completed and the bus is currently in idle status, each node detects a gap length of a predetermined idle period (e.g., sub-action gap) set in each transfer mode, and it determines whether or not the bus is currently in idle status based on the detection result.

At step S401, the node determines whether or not a predetermined gap length corresponding to asynchronous data or isochronous data to be transferred has been detected. So far as the node has not detected the predetermined gap length, it cannot request a bus use right to start data transfer, accordingly, the node waits until the predetermined gap length has been

detected.

When the predetermined gap length has been detected at step S401, the node determines whether or not there is data to be transferred at step S402. If YES, it issues a signal requesting a bus use right to the root at step S403. As shown in Fig. 10A, this signal requesting the bus use right is relayed by the respective devices in the network, and forwarded to the root. If it is determined at step S402 that there is no data to be transferred, the process returns to step S401.

At step S404, if the root has received a single or plurality of request signals for the bus use right, it examines the number of nodes requesting the bus use right at step S405. From the determination at step S405, if the number of the nodes requested the bus use right is one, that node is provided with bus use permission immediately after the requirement. On the other hand, if the number of the nodes is more than one, arbitration is performed to determine one node to be provided with the bus use right immediately after the requirement. The arbitration does not always provide a bus use right to the same node, but equally provides a bus use right to the respective nodes (fair arbitration).

The processing at the root branches at step S407

into processing for the node dominated in the arbitration at step S406, and processing for the other nodes lost in the arbitration. In a case where there is one node that requested the bus use right, or one node has dominated in the arbitration, the node is provided with an permission signal indicative of bus use permission at step S408. The node starts data (packet) transfer immediately after it receives the permission signal (step S410). On the other hand, the nodes lost in the arbitration receive a DP (data prefix) packet indicative of rejection of the bus use request at step S409. The processing for the node that received the DP packet returns to step S401 to request a bus use right again. Also, the processing for the node that completed data transfer at step S410 returns to step S401.

[Transaction Layer]

The transaction layer includes a read transaction, a write transaction and a lock transaction.

In a read transaction, an initiator (requiring node) reads data from a specific address in the memory of a target (response node). In a write transaction, the initiator writes data into a specific address of the memory of the target. In a lock transaction, the

initiator transfers reference data and update data to the target. The reference data is combined with data of the address of the target, into a designation address to specify a specific address of the target. Data at the designation address is updated by the update data.

Fig. 27A shows a request-response protocol with read, write and lock commands, based on the CSR architecture in the transaction layer. In Fig. 27A, the request, notification, response and confirmation are service units in the transaction layer 814.

A transaction request (TR_DATA.request) is packet transfer to a response node; a transaction indication (TR-DATA.indication) is notification of arrival of the request to the response node; a transaction response (TR_DATA.response) is transmission of acknowledgment; and a transaction confirmation (TR_DATA.confirmation) is reception of acknowledgment.

[Link Layer]

Fig. 27B shows services in the link layer 812. The services are service units of a link request (LK_DATA.request) to require packet transfer from the response node, a link indication (LK_DATA.indication) indicating packet reception to the response node, a link

response (LK_DATA.response) as acknowledgment transmitted from the response node, a link confirmation (LK_DATA.confirmation) as confirmation of the acknowledgment transmitted from the response node. One packet transfer process is called a sub-action including an asynchronous sub-action and an isochronous sub-action. Hereinbelow, the respective operations of the sub-actions will be described.

[Asynchronous Sub-action]

The asynchronous sub-action is asynchronous data transfer.

Fig. 12 shows transition in the asynchronous transfer. In Fig. 12, the first sub-action gap represents the idle status of the bus. At a point where the idle time has become a predetermined value, a node which is to perform data transfer requests a bus use right, then bus arbitration is executed.

When the use of bus has been allowed by the arbitration, data in the form of packet is transferred, and a node which receives the data sends a reception acknowledgment code ACK as a response, or sends a response packet after a short gap called ACK gap, thus the data transfer is completed. The code ACK comprises

4-bit information and a 4-bit checksum. The code ACK, including information indicative of success, busy or pending status, is immediately sent to the data-sender node.

Fig. 13 shows a packet format for asynchronous transfer. The packet has a data area, a data CRC area for error correction, and a header area in which a destination node ID, a source node ID, a transfer data length and various codes are written.

The asynchronous transfer is one-to-one communication from a sender node to a receiver node. A packet sent from the sender node is relayed by the respective nodes in the network, however, as these nodes are not designated as the receiver of the packet, they ignore the packet, then only the receiver node designated by the sender node receives the packet.

[Split Transaction]

The services in the transaction layer 814 are performed as a set of transaction request and transaction response, as shown in Fig. 27A. if the processings in the link layer 812 and the transaction layer 814 of the target (response node) are performed at a sufficiently high speed, the request and the response

are not performed as independent sub-actions but as one sub-action in the link layer 812. However, if the processing speed of the target is low, the request and the response must be processed by independent sub-actions. This is called a split transaction.

Fig. 28A shows an operation example of the split transaction. In Fig. 28A, in response to a write request from a controller as an initiator (requiring node), a target returns a pending. The target returns confirmation information to the write request from the controller, thus gains time for data processing. When a sufficient period for the data processing has elapsed, the target sends a write response to the controller, thus completes the write transaction. Note that another node can perform the operation of the link layer 812 between the request and response sub-actions.

Fig. 28B shows transitional statuses of transfer in case of the split transaction. In Fig. 28B, a sub-action 1 represents the request sub-action; and a sub-action 2, the response sub-action.

In the sub-action 1, the initiator sends a data packet indicative of the write request to the target, and the target receives the data packet, returns "pending" indicative of the confirmation of the above

information, as an acknowledge packet. Then, the request sub-action is completed.

Then, when a sub-action gap has been inserted, the target sends a write response as a data packet with no data, in the sub-action 2. The initiator receives the data packet, returns a "complete" response as an acknowledge packet. Then, the response sub-action is completed.

Note that the period from the completion of the sub-action 1 to the beginning of the sub-action 2 can be minimized to a period corresponding to the sub-action gap, while maximized to a period corresponding to a maximum waiting period set in the nodes.

[Isochronous Sub-action]

Isochronous transfer, which can be regarded as the greatest feature of the 1394 serial bus is appropriate to multimedia data transfer which requires realtime transfer of, especially, AV data.

Further, the asynchronous transfer is one-to-one transfer, whereas the isochronous transfer is broadcasting transfer from one sender node to all the other nodes.

Fig. 14 shows transition in the isochronous

transfer. The isochronous transfer is executed on the bus in a predetermined cycle, called "isochronous cycle". The period of the isochronous cycle is 125 μs. A cycle start packet (CSP) indicates the start of the isochronous cycle for synchronizing the operations of the respective nodes. When data transfer in a cycle has been completed and a predetermined idle period (sub-action gap) has elapsed, a node which is called "cycle master" sends the CSP indicative of the start of the next cycle. That is, this interval between the issuance of CSP's is 125 μs.

As channel A, channel B and channel C in Fig. 14, the respective packets are provided with a channel ID, so that plural types of packets can be independently transferred within one isochronous cycle. This enables substantially-realtime transfer among the plural nodes. The receiver node can receive only data with a predetermined channel ID. The channel ID does not indicate an address of the receiving node, but merely indicates a logical number with respect to the data. Accordingly, one packet sent from a sender node is transferred to all the other nodes, i.e., broadcasted.

Similar to the asynchronous transfer, bus arbitration is performed prior to the packet

broadcasting in isochronous transfer. However, as the isochronous transfer is not one-to-one communication like the asynchronous transfer, the reception acknowledgment code ACK used as a response in the asynchronous transfer is not used in the isochronous transfer.

Further, an isochronous gap (iso gap) in Fig. 14 represents an idle period necessary for confirming prior to isochronous transfer that the bus is in idle status. If the predetermined idle period has elapsed, bus arbitration is performed with respect to node(s) desiring isochronous transfer.

Fig. 15A shows a packet format for isochronous transfer. Various packets divided into channels respectively have a data field, a data CRC field for error correction and a header field containing information such as a transfer-data length, a channel No., various codes and error-correction header CRC as shown in Fig. 15B.

[Bus Cycle]

In practice, both isochronous transfer and asynchronous transfer can be mixedly performed on the 1394 serial bus. Fig. 16 shows transition in the

isochronous transfer and asynchronous transfer mixedly performed on the 1394 serial bus.

The isochronous transfer is performed prior to the asynchronous transfer because after the CSP, the isochronous transfer can be started with a gap (isochronous gap) shorter than the idle period necessary for starting the asynchronous transfer. Accordingly, the isochronous transfer has priority over the asynchronous transfer.

In the typical bus cycle as shown in Fig. 16, upon starting the cycle #m, a CSP is transferred from the cycle master to the respective nodes. The operations of the respective nodes are synchronized by this CSP, and node(s) that waits for a predetermined idle period (isochronous gap) to perform isochronous transfer participates in bus arbitration, then starts packet transfer. In Fig. 16, a channel e, a channel s and a channel k are transferred by the isochronous transfer.

The operation from the bus arbitration to the packet transfer is repeated for the given channels, and when the isochronous transfer in the cycle #m has been completed, the asynchronous transfer can be performed. That is, when the idle period has reached the sub-action gap for the asynchronous transfer, node(s) that is to

perform the asynchronous transfer participates in bus arbitration. Note that only if the sub-action gap for starting the asynchronous transfer is detected, after the completion of isochronous transfer and before the next timing to transfer the CSP (cycle synch), the asynchronous transfer can be performed.

In the cycle #m in Fig. 16, the isochronous transfer for three channels is performed, and then two packets (packet 1 and packet 2) including ACK are transferred by the asynchronous transfer. When the asynchronous packet 2 has been transferred, as the next cycle synch point to start the subsequent cycle m+1 comes, the transfer in the cycle #m ends. Note that during the asynchronous or isochronous transfer, if the next cycle synch point to transfer the next CSP has come, the transfer is not forcibly stopped but continued. After the transfer has been completed, a CSP for the next cycle is transferred after a predetermined idle period. That is, when one isochronous cycle is continued for more than 125 μs, the next isochronous cycle is shorter than the reference period 125 μs. In this manner, the isochronous cycle can be lengthened or shortened based on the reference period 125 μs.

However, it may be arranged such that the isochronous transfer is performed in every cycle, while the asynchronous transfer is sometimes postponed until the next cycle or the cycle further subsequent to the next cycle, so as to maintain realtime transfer. The cycle master also manages information on such delay.

[FCP]

In an AV/C protocol, a Function Control Protocol (FCP) is provided to control devices on the 1394 serial bus. For transmission of control commands and responses in the FCP protocol, an asynchronous packet defined by the IEEE 1394 standards is employed. In the FCP protocol, a node on the controller side is called a controller, and a node on the controlled side, a target. An FCP packet frame sent from the controller to the target is called an AV/C command frame; an FCP packet frame returned from the target to the controller, an AV/C response frame.

Fig. 29 shows a case where a node A is a controller and a node B is a target. In the register address provided for the respective nodes, 512 bytes from "0x0000B00" are assigned to a command register; and 512 bytes from "0x0000D00" are assigned to a response

register. Data is written into the register of a designated address by a packet frame using the asynchronous transfer. The relation between the transmission of the AV/C command frame by the controller and the response with the AV/C response frame by the target is called an AV/C transaction. In a general AV/C transaction, a target which has received a command frame must send a response frame to a controller within 100 ms.

Fig. 30 shows the packet format for the asynchronous transfer including an FCP packet frame. A command frame or a response frame is inserted into a data area of the asynchronous data packet as shown in Fig. 15A, and the AV/C transaction is performed.

Fig. 31 shows the structure of the AV/C command frame. Fig. 32 shows the structure of the AV/C response frame. As the FCP packet frame, an FCP data part is arranged after "ctype", "response", "subunit_type" and "subunit_ID" in the header.

"ctype" indicates a command type in the command frame, with status "CONTROL", "STATUS", "INQUIRY" or "NOTIFY".

"response" indicates a response code in the response frame, with status "ACCEPTED", "REJECTED", "IN_TRANSITION", "IMPLEMENTED", "CHANGED" or "INTERIM".

Further, "subunit_type" indicates the classification of a device, and "subunit_ID" indicates an instance number.

The FCP data part has an operation code (opcode) + operand (oprand) structure. The target is controlled and the AV/C response is performed by using various AV/C commands.

The operation code (opcode) in the command frame as shown in Fig. 31 is one of commands as shown in Fig. 41. The respective commands are performed with contents according to the command types set at the "ctype".

A command where the "ctype" designates the status "CONTROL" is a control command used for controlling the target device or setting the target to the content set after the operand (oprand). A command where the "ctype" designates the status "STATUS" is used for obtaining a status corresponding to the command. A command where the "ctype" designates the status "INQUIRY" is used for inquiring about contents which can be set by the command. A command where the "ctype" designates the status "NOTIFY" is used for performing confirmation of the command.

In each command, necessary content is set at the operand, and the command is written into the command

frame.

In the operation code of the response frame, one of response codes as shown in Fig. 41 is set. Each response has an operand corresponding to the response. As information indicating whether the execution of the command has been normally completed or ended with error, is set at the operand, error processing can be performed in accordance with the operand.

[Communication Using LOGIN Protocol]

Fig. 17 shows the interface of the 1394 serial bus in comparison with respective layers of an OSI model often used in a LAN. In the OSI model, a physical layer 1 and a data link layer 2 respectively correspond to a physical layer 811 and a link layer 812 (both shown in Fig. 2) in a lower layer 4 of the 1394 serial bus interface. In the 1394 serial bus interface, a transport protocol layer 5 and a presentation layer 6 as upper layers correspond to an upper layer 3 of OSI model including a network layer, a transport layer, a session layer and a presentation layer. Further, a LOGIN protocol 7, operates between the lower layer 4 and the transport protocol layer 5 of the 1394 serial bus interface.

In Example 1 in Fig. 17, by providing the LOGIN protocol 7 to a device based on a serial bus protocol (SBP-2) 8 for a peripheral device such as a printer, the peripheral device uses a protocol based on the protocol SBP-2 to notify a target device of data transfer with the target device. In Example 2, with respect to a device protocol 9 specialized on the 1394 serial bus interface, by providing the LOGIN protocol 7 to respective devices, the devices can determine whether or not the target device supports their protocol, with each other.

Fig. 18 shows the basic operation of the LOGIN protocol. When a printer device executes a print task 10 from a host device, the printer device first selects one of printer protocols A to C for data communication, based on communication by the LOGIN protocol 7. Thereafter, the printer device performs print data transfer in accordance with the selected printer protocol. That is, upon connection between the printer device which supports a plurality of printer protocols and a host device, the printer device first judges the transport protocol 5 of the host device based on the LOGIN protocol 7, selects a printer protocol corresponding to the transport protocol 5 of the host

device, and performs transfer/reception of print data or commands in accordance with the selected printer protocol, thus performs the print task 10.

Fig. 19 shows connection status in the 1394 serial bus, where devices (PC12, scanner 13 and VCR 14 etc.) with the LOGIN protocol 7 are connected to a printer 11 corresponding to plurality of printer protocols. The printer 11 can process print tasks from the respective devices by changing the printer protocol in accordance with the transport protocol 5 of a device that requests connection with the printer device.

Fig. 20 shows the flow of log-in operation.

At STEP 1:

· The host device locks a target device (a multi-protocol printer in this case).

· The target device examines the capability of the host device (including the transport protocol). Note that the capability has been stored into a capability register 503 (to be described later) of the host device.

· The target device sets the capability (including the transport protocol) of the host device.

At STEP 2:

· Print data is transferred by the protocol determined at the STEP 1.

At STEP 3:

· The host device disconnects the connection with the target device.

Fig. 21 shows control/status register (CSR) which is prepared by a printer as the target device so that the LOGIN protocol is installed, including a lock register 501, a protocol register 502 and the capability register 503. These registers are provided in predetermined addresses in initial unit space in the address space of the 1394 serial bus. That is, as shown in Fig. 3, within the 48-bit address area provided to the devices, "0xFFFFF" in the first 20-bits is called "register space", wherein a register (CSR core) as the core of the CSR architecture is arranged in the first 512 bytes. Note that information common to devices connected via the bus is provided in this register space. Further, "0-0xFFFFD" is called "memory space", and "0xFFFFE", "private space". The private space is an address which can be freely used in the device for communication among the devices.

The lock register 501 indicates a locked status of a resource, with a value "0" indicative of log-in enable status, and any value but "0", an already-logged-in and

locked status. The capability register 503 indicates a protocol where each bit represents a protocol, with a value "1" bit indicating that a corresponding protocol can be set, while a value "0" bit, that a corresponding protocol cannot be set. The protocol register 502 indicates a currently set protocol. That is, each bit of the protocol register 502 corresponds to each bit of the capability register 503, and the value of a bit of the protocol register 502 corresponding to the set protocol is "1".

Fig. 22 is a flowchart showing LOGIN processing in the host device.

To start the LOGIN processing, first, the data of the lock register 501, the protocol register 502 and the capability register 503 of a target device e.g., a printer, to be logged-in, is checked by a read transaction. At this time, from the data of the capability register 503, it is examined whether or not the target device supports a protocol used by the host device for communication (step S601). If the target device does not support the protocol of the host device, the LOGIN processing is terminated at step S602.

Further, if the data value of the lock register 501 is not "0", it is determined that another device is

in logged-in status, and the LOGIN processing is terminated. If the data value of the lock register 501 is "0", it is determined that log-in is currently possible (step S602).

In case of log-in enable status, the process moves to resource lock processing where the log-in is set by writing "1" into the lock register 501 of the printer by using a lock transaction (step S603). The target device is locked in this status, and it is uncontrollable from other devices and the register values cannot be changed.

As described above, in the status where the resource of the target device is locked, protocol setting is performed next. As the printer as the target device of the present embodiment supports a plurality of printer protocols, the printer must be informed of the protocol which can be used by the host device before it receives print data. In the present embodiment, the protocol to be used is notified to the printer by setting the corresponding bit of the protocol register 502 of the printer by a write transaction (step S604).

At this point, as the protocol used by the host device for communication has been notified to the target device and the target device is in locked status, the host device that is currently logged in the target

device performs data (print data in this case) transfer (step S605).

As the data transfer has been completed, the host device logs out from the printer by clearing the lock register 501 and the capability register 503 of the target device (step S606).

Fig. 23 is a flowchart showing the LOGIN processing in the printer as the target device.

The printer generally waits for log-in from a host device. As a print request from a host device is started by reading data values from the lock register 501, the protocol register 502 and the capability register 503 of the printer, the registers must be in read-enable status. This processing will be described on the assumption that a host device which is to perform printout has locked the printer (step S701).

The printer waits for notification of an available protocol from the host device (step S702). The printer receives the notification of available protocol in locked status so as to maintain the protocol register 502 unchanged by another device's request in mid-course of the log-in processing.

When the available protocol has been assigned (step S703), the printer switches its own protocol to

the notified protocol (steps S704, S706 and S708), and performs communication in accordance with the protocol of the host device (steps S705, S707 and S709).

When the communication has been completed, the printer confirms that the lock register 501 and the capability register 503 have been cleared (step S710), and returns to the log-in waiting status (step S701).

[Example in Consideration of Device without LOGIN Protocol]

Fig. 24 is an explanatory view showing a case in consideration of a device without the LOGIN protocol 7. Compared with the first example as shown in Fig. 18, this example is applicable to a device having a protocol D in which the LOGIN protocol 7 is not installed. That is, to assure the device only corresponding to the conventional protocol D (e.g., AV/C protocol) of print operation, as well as devices having the LOGIN protocol 7, the printer side has the protocol D.

In this case, if the printer recognizes, by a print request performed at the beginning of connection, that the host device does not correspond to the LOGIN protocol 7, the printer tries communication with the host device by using the protocol D, and if the

communication can be established, the printer executes the print task 10 in accordance with the protocol D.

Fig. 25 shows the IEEE 1394 serial interface according to this example in comparison with the OSI model. Example 3 uses, as a model, an AV device 15 based on the AV/C protocol. In the AV device 15, the LOGIN protocol 7 is not installed. Example 4 uses, as a model, a scanner 16, in which the LOGIN protocol 7 is not installed, but a non-standard protocol for scanner is installed.

That is, regarding a device in which the LOGIN protocol 7 is not installed, if the printer can perform communication using the protocol of the device, the printer can perform print task from the device. This increases the types of devices that can use the printer.

[Direct Print Control]

Hereinbelow, print procedures in the printer and the image providing device will be described. In this case, a Direct Print Protocol (DPP), is employed as a protocol to directly connect the printer and the image providing device, and enable the printer to form an image based on image data provided from the image providing device.

The DPP protocol basically comprises a command register (command) for writing a command in the initial unit space (the unit space in Fig. 3), a response register (response) for writing a response to the command, a data register (data) for writing transfer data, and a format register (format) for handling format information corresponding to data format for each transfer data.

Fig. 33 shows a register map in which the command register and the response register are the same as those described in Fig. 29. In the following description, the image providing device as a controller provides image data to the printer as a target and an image is printed based on the image data.

A command register 42-1, arranged from a fixed address "0xFFFFF0000B00" on the printer side, has a 512 byte memory space into which the image providing device writes various commands for the printer. If the image providing device side also has a command register, the printer can write commands into the register. The command written into the command register is called a command frame.

A response register 42-2, arranged from fixed address "0xFFFFF0000D00", has a 512 byte memory space

into which a response is written with respect to the various commands written from the image providing device to the printer. If the printer side also has a response register, the image providing device can write a response into the register. The response written into the response register 42-2 is called a response frame. Note that in Fig. 29, the upper address part "0xFFFF" is omitted.

A data register 42-3, having a default address "FFFFF0003000h", is set at an arbitrary effective address by commands "BlockAddress" and "BufferConfig" (commands to define the address of the data register). The memory space of the data register 42-3 is set with a range predetermined by commands "BlockSize" and "BufferConfig" (commands to define the memory space of the data register). The data register 42-3 is used for data transfer between the image providing device and the printer. Upon printing, the image providing device writes print data for the printer into this register. The print data is based on a pre-set image format. The data written into the data register 42-3 is called a data frame.

A format register 42-2 is a set of registers corresponding to various data formats to be described

later. The respective registers are used for setting format information necessary for the respective data formats. That is, the format register 42-2 is used for writing the format information for the printer from the image providing device. The format information written into the format register 42-2 is called a format frame.

Fig. 34 shows the flow of frames from the image providing device to the printer, i.e., shows the flow of the command frame, the response frame, data frame and the format frame. The printer performs printing in accordance with the frames outputted from the image providing device.

A command sent from the image providing device to the printer is written as a command frame into a command register 43-4 of the printer. The command is used for printing, as shown in Fig. 41. A response to this command is written by the printer into a response register 43-2 of the image providing device. The response includes information indicating whether or not the command has been properly executed, a return value to the command and the like. Print data sent from the image providing device to the printer is written into a data register 43-6 of the printer. Format information sent from the image providing device to the printer is

written as a format frame into a format register 43-7 of the printer.

Fig. 35 shows the construction of the format register 43-7. The format register 43-7 includes a read only register INQUIRY 44-1 for inquiry and a read/write register CONTROL/STATUS 44-2 for setting and information acquisition. The registers INQUIRY 44-1 and the CONTROL/STATUS 44-2 comprise register groups of the same structure. That is, the INQUIRY 44-1 has registers 44-3 to 44-7, while the CONTROL/STATUS 44-2 has registers 44-9 to 44-13.

More specifically, the format register group includes the registers 44-3 and the 44-4 (44-9 and 44-10) constituting a print common register group, and the registers 44-5 to 44-7 (44-11 to 44-13) constituting a printer format register group.

The common register group, which is a group of registers common to all the data formats, has the register GLOBAL 44-3 (44-9) common to all the printers and the register LOCAL 44-4 (44-10) unique to each printer.

The printer format register group is a group of n registers unique to the respective data formats, i.e., the register format[1] 44-5 (44-11) to the register

format[n] 44-7 (44-13). The registers format[1] to format[n] correspond to data formats to be described later. One of the printer format register group is assigned to each installed data format.

Note that the address of each format register is provided to the image providing device as a response to a command for setting a data format.

Fig. 36 shows the detailed construction of the status register 44-8 of the common register group. The status register 44-8 comprises a 32-bit common status register 45-1 holding a status common to the respective vendor printers and a 32-bit vendor specific status register 45-2 holding a status defined by each vendor. Further, the extension of the vendor specific status register 45-2 is defined as follows, by a v flag (vendor status available flag) at the 31th bit of the common status register 45-1:

v flag: "0" = not available
"1" = available

Further, information held in the common status register 45-1 is as follows:

error.warning: status of error, warning and the like

paper state: status on print sheet

print state: status on printing situation

Fig. 37 shows the details of information held in the register GLOBAL 44-3 (44-9) of the common register group. The register GLOBAL 44-3 (44-9) holds information common to all the printers having the DPP (Direct Print Protocol), i.e., information which does not differ in printer type. Fig. 37 shows an example of the information, including media-type 46-1 indicative of the type of print medium, paper-size 46-2 indicative of a print sheet size, page-margin 46-3 indicative of a page margin value, page-length 46-4 indicative of a page length, page-offset 46-5 indicative of page offset, print-unit 46-6 indicative of printer unit information, color-type 46-7 indicative of the color type of printer, bit-order 46-8 indicative of the bit order of data, and the like.

Fig. 38 shows the details of information held in the register LOCAL 44-4 (44-10) of the common register group. The register LOCAL 44-4 (44-10) holds information unique to each of the printers having the DPP protocol,

i.e., information which differs in printer type. Fig. 38 shows an example of the information, including paper 47-1 indicative of the type of print sheet of a printer, CMS 47-2 indicative of a color matching method, ink 47-3 indicative of ink type of the printer, e.g., an ink-jet printer.

Fig. 39 shows the details of information held in the register format[1] 44-5. In Fig. 39, information on EXIF (Exchangeable Image File Format) as one image data format is held. The information includes inX-rate 48-1 indicative of the rate of X-directional input, inY-rate 48-2 indicative of the rate of Y-directional input, outX-rate 48-3 indicative of the rate of X-directional output, and outY-rate 48-4 indicative of the rate of Y-directional output. In this case, printing is performed by enlarging/reducing given EXIF image data in the XY directions in accordance with the respective contents of the register.

Fig. 40 shows the details of information held in the register format[2] 44-6. In Fig. 40, information on Raw RGB as one image format is held. The Raw RGB format has a structure to represent each pixel by R (red), G (green) and B (blue) data. The information includes inX-rate 49-1 indicative of the rate of X-directional input,

inY-rate 49-2 indicative of the rate of Y-directional input, outX-rate 49-3 indicative of the rate of X-directional output, outY-rate 49-4 indicative of the rate of Y-directional output, XY-size 49-5 indicative of an XY-directional fixed pixel size, bit-pixel 49-6 indicative of the number of bits per pixel, X-size 49-7 indicative of the number of pixels in the X direction, Y-size 49-8 indicative of the number of pixels in the Y direction, plane 49-9 indicative of the number of color planes, X-resolution 49-10 indicative of X-directional resolution, Y-resolution 49-11 indicative of Y-directional resolution, pixel-format 49-12 indicative of pixel type, and the like. In this case, printing is performed by enlarging/reducing given Raw RGB format image data and/or changing the resolutions in the XY directions in accordance with the respective contents of the register, further, processing the image data based on image size information and the like.

Fig. 41 shows commands and responses to the commands. The commands are classified based on several types, i.e., "status" type commands relating to status, "control" type commands for printer control, "block/buffer" type commands for data transfer setting, "channel" type commands for channel setting, "transfer"

type commands relating to a transfer method, "format" type commands relating to format setting, "login" type commands relating to log-in, "data" type commands relating to data transfer, and the like. Note that the log-in, "login" type commands as aforesaid, and a command Login, a response LoginResponse, a command Logout and a response LogoutResponse described later are unrelated to the LOGIN protocol as aforementioned.

More specifically, the "status" type commands include a command GetStatus to obtain the status of a printer and its response GetStatusResponse 50-1 and the like.

The "control" type commands include a command PrintReset to reset the printer and its response PrintResetResponse 50-2, a command PrintStart to instruct to start printing and its response PrintStartResponse 50-3, a command PrintStop to instruct to stop printing and its response PrintStopResponse 50-4, a command InsertPaper to instruct to supply a print sheet and its response InsertPaperResponse 50-5, a command EjectPaper to instruct to discharge a print sheet and its response EjectPaperResponse 50-6, a command CopyStart to instruct to start copying image data and its response CopyStartResponse 50-7, a command

CopyEnd to instruct to terminate copying image data and its response CopyEndResponse 50-8, and the like.

The "block/buffer" type commands include a command BlockSize to designate a block size and its response BlockSizeResponse 50-9, a command BlockAddress to designate a block address and its response BlockAdressResponse 50-10, a command FreeBlock to obtain the number of available blocks and its response FreeBlockResponse 50-11, a command WriteBlocks to instruct to write data into blocks and its response WriteBlocksResponse 50-12, a command BufferConfig to designate buffer information and its response BufferConfigResponse 50-13, a command SetBuffer to designate to start obtaining data from buffer and its response SetBufferResponse 50-14, and the like.

The "channel" type commands include a command OpenChannel to open a channel and its response OpenChannelResponse 50-15, a command CloseChannel to close the channel and its response CloseChannelResponse 50-16, and the like.

The "transfer" type commands include a command TransferMethod to designate a data transfer method and its response TransferMethodResponse 50-17 and the like.

The "format" type commands include a command

SetFormat to set a format and its response SetFormatResponse 50-18 and the like.

The "login" type commands include a command Login to perform log-in and its response LoginResponse 50-19, a command Logout to perform log-out and its response LogoutResponse 50-20, a command Reconnect to perform reconnection and its response ReconnectResponse 50-21, and the like.

These commands are written into a command frame.

Further, the "data" type commands include commands WriteBlock 50-22 and WriteBuffer 50-23 to write data, a command PullBuffer 50-24 to read data, and the like. These commands are written/read with respect to a data frame, and there have no responses corresponding to these commands.

The image providing device sets a value corresponding to each of the various commands as shown in Fig. 41 at the operation code (opcode), and writes the command into the command register 43-4 of the printer. Thus, the command is executed by the printer. The response to the command has the same value as that of the command. The printer sets the response at the operation code of the response frame as shown in Fig. 32, and writes the frame into the response register 43-2 of

the image providing device. By the response, the image providing device receives the result of execution of the command.

Fig. 42 shows the image data formats supported by the DPP protocol. The printer must support Raw image data of at least one of these formats. Further, the printer can support other plural formats as optional formats.

Fig. 43 shows the flow of format setting processing. First, the image providing device sends the command SetFormat (INQUIRY) to the printer at step S500, and the printer returns the SetFormatResponse at step S501. The image providing device obtains the address of the INQUIRY register of the printer by the returned response.

Next, the image providing device sends the command SetFormat (CONTROL/STATUS) to the printer at step S502, and the printer returns the SetFormatResponse at step S503. The image providing device obtains the address of the CONTROL/STATUS register of the printer by the returned response.

The image providing device reads the INQUIRY register of the printer at steps S504-1 to S504-m, and obtains the format supported by the printer and set

items of the format. Next, the image providing device reads the STATUS/CONTROL register of the printer at steps S505-1 to S505-n, obtains the set values of the format, then writes data into the STATUS/CONTROL register of the printer, thus sets the format.

The data transfer in the DPP protocol uses the following two packets.

- Control command packet for flow control
- Packet for data transmission

In the present embodiment, the following five types of data transfer methods are used in accordance with the difference among data transmission methods and flow controls. In any method, control commands for the flow control are based on the FCP protocol. However, The control commands are not limitted to the FCP protocol.

Transfer method 1: Response model
Transfer method 2: Simplified response model
Transfer method 3: PUSH large buffer model
Transfer method 4: PULL buffer model
Transfer method 5: Isochronous model

In actual transfer, one of the above methods is selected and set in a procedure similar to the format setting procedure as shown in Fig. 43. Note that as shown in Fig. 44, the command TransferMethod and the response TransferMethodResponse are employed.

Fig. 44 shows the flow of data-transfer method setting processing. The image providing device obtains a currently available data transfer method of the printer by the command TransferMethod and the response TransferMethodResponse in the command type INQUIRY (steps S600-1 and S600-2), and obtains and sets a data transfer method currently set at the printer, by the command TransferMethod and the response TransferMethodResponse of the command type CONTROL/STATUS (steps S600-3 and S600-4).

In any of the above five type of transfer methods, the control commands for flow control are based on the FCP protocol as a protocol for controlling a device on the 1394 serial bus. The transfer of control command by the FCP protocol is always performed by an asynchronous write transaction, in both transmission and response.

Fig. 45 shows a register map of registers necessary for the transfer method 1 and the transfer

method 2, in address space of the 1394 serial bus. In the present embodiment, a node on the controller side is the image providing device (controller in the FCP protocol), and a node on the controlled side is the printer (target in the FCP protocol).

Both image providing device and printer have a command register (601-1 and 601-7) with offset of 512 bytes from "0x0B00" and a response register (601-2 and 601-8) with offset of 512 bytes from "0x0D00" in the register space. These registers are based on the FCP protocol and the AV/C commands.

The flow control is made by writing the command frame 601-4 and the response frame 601-5 into the command register (601-1 and 601-7) and the response register (601-2 and 601-8), respectively. Further, for print data transfer, a dedicated data frame is defined. That is, a data register (601-3 and 601-9) for a block size is provided from offset "0x3000" in the register space, and print data is transferred by writing a data frame 601-6 into the data register (601-3 and 601-9). Note that the block size is 512 bytes, for example.

Fig. 46 shows a data packet frame comprising a data header 602-1, a data payload 602-2 and the like.

Fig. 47 shows the structure of the data header

602-1. In the data header 602-1, upper 8 bits are assigned to a block count area 603-1, and lower 8 bits, to a future reserve area 603-2. The block count area 603-1 is internally used by the target (printer) to count the number of blocks transferred at one block transfer. Note that as the block count area 603-1 has 8 bits, it takes a value "0" to "255".

Fig. 48 shows data packet frame processing in the printer in block transfer. A data packet received by the printer is first written into a data register 604-1 of the printer. The printer has buffers (604-2 to 604-5) of the same size of that of the data packet. The data packet written in the data register 604-1 is sequentially moved to these buffers (604-2 to 604-5). Note that the number of the data buffers is preferably 255, the maximum block count value, but it may be a value less than 255. The respective buffers correspond to block count values. A data packet when the block count value is "0" is stored into a buffer of Block[0]; and a data packet when the block count value is "1" is stored into a buffer of Block[1]. The data header 602-1 is removed from the data packets stored in the buffers (604-2 to 604-5), and the data packets are developed in a memory space 604-6 of the printer.

[Transfer Method 1]

The transfer method 1 as the response model defines a data packet frame for data transmission, provides a data register, performs flow control by control commands, while transfers print data by a write transaction.

Fig. 49 shows commands and responses FreeBlock in the transfer method 1. In the transfer method 1, prior to data transfer, the image providing device uses the commands and responses FreeBlock to obtain information indicating the number of data packets to be transferred.

The image providing device transfers the FreeBlock command by a write transaction (step S605-1), and the printer returns an ACK packet indicative of the acknowledgment of the transaction (step S605-2). The printer returns the FreeBlockResponse to notify a FreeBlockCount (step S605-3) which is the number of currently available blocks, and the image providing device returns an ACK packet indicative of the acknowledgment of the transaction (step S605-4).

Fig. 50 shows the flow of data transfer in the transfer method 1. The image providing device logs in the printer by the command and response Login of the DPP

protocol (steps S606-1 and S606-2), and sets a format to be used for data transfer by using the above-described format register group (step S606-3). Thereafter, the image providing device opens a logic channel by the command and response OpenChannel (steps S606-4 and S606-5). Next, data transfer is started, and print data is transferred in the data packet format as shown in Fig. 46. Further, the packet transfer is performed in correspondence with the number of blocks the same as the number of data buffers on the printer side, as shown by references 604-2 to 604-5 in Fig. 48, as one cycle.

In the transfer method 1, the print data is transferred as follows. The image providing device obtains the FreeBlockCount of the printer by the command and response FreeBlock (steps S606-6 and S606-7), and sequentially transfers data packets of the same number as that of the FreeBlockCount, by the command WriteBlock (step S606-8). Note that the command WriteBlock is used for transferring print data packets from the data register 601-3 of the image providing device to the data register 601-9 of the printer. Note that there is no response corresponding to the command WriteBlock, and the printer returns an ACK packet to confirm that the data packets have been stored into the data register

601-9.

Then, block transfer is performed by repeating packet transfer, with the commands WriteBlock of the same number of the FreeBlockCount and ACK packets corresponding to the commands until all the series of print data has been outputted from the image providing device, and between the respective block transfers, the FreeBlockCount of the printer is obtained by the command and response FreeBlock.

When the print data transfer has been completed, the image providing device closes the logic channel by the command and response CloseChannel (steps S606-10 and S606-11), and logs out from the printer by the command and response Logout of the DPP protocol (steps S606-12 and S606-13).

[Transfer Method 2]

The transfer method 2 as the simplified response model performs data transfer in the same procedure as that of the transfer method 1 except the method to obtain the FreeBlockCount.

Fig. 51 shows the flow of data transfer in the transfer method 2. The image providing device logs in the printer by the command and response Login of the DPP

protocol (steps S607-1 and S607-2), and sets a format to be used for data transfer by using the above-described format register group (step S607-3). Thereafter, the image providing device opens a logic channel by the command and response OpenChannel (steps S607-4 and S607-5). Next, data transfer is started, and print data is transferred in the data packet format as shown in Fig. 46. Further, the packet transfer is performed in correspondence with the number of blocks the same as the number of data buffers on the printer side, as shown by references 604-2 to 604-5 in Fig. 48, as one cycle.

In the transfer method 2, the print data is transferred as follows. The image providing device obtains the FreeBlockCount of the printer of the printer by the command WriteBlocks and response WriteBlockResponse (steps S607-6 and S607-7). Note that the response at step S607-7 is of the INTERIM type for performing the acquisition of the FreeBlockCount only by the response from the printer side. The image providing device sequentially transfers data packets of the same number as the obtained FreeBlockCount, by the command WriteBlock (step S607-8), and the printer returns the above-described ACK packet (607-9). Then, block transfer is performed by repeating packet transfer by the

commands WriteBlock of the same number as the FreeBlockCount and ACK packets corresponding to the commands until all the series of print data has been outputted from the image providing device. Note that at the second cycle and the subsequent cycles in the block transfer, the FreeBlockCount is notified to the image providing device by the WriteBlocksResponse from the printer, at the end of each block transfer cycle (step S607-10). The WriteBlocksResponse is of the CONTINUE type used for continuing the acquisition of the FreeBlockCount only by the response from the printer.

When the print data transfer has been completed, the image providing device closes the logic channel by the command and response CloseChannel (steps S607-11 and S607-12), and logs out from the printer by the command and response Logout of the DPP protocol (steps S607-13 and S607-14).

[Method to Obtain FreeBlockCount]

Hereinbelow, the method to obtain the FreeBlockCount, which is the difference between the transfer method 1 and the transfer method 2, will be described in detail.

Fig. 52 shows the commands and responses FreeBlock

in the transfer method 1 at steps S606-6 and S606-7, in detail, including the ACK packets of the write transaction omitted in Fig. 50. In this case, both initiator device (image providing device) and the target device (printer) perform processing of the link layer and transaction layer at a relatively low speed.

When the image providing device writes the command FreeBlock into the command register by a write transaction (step S608-1), the above-described ACK packet indicative of "pending" is returned from the link layer of the printer (step S608-2). Next, the image providing device sends a command FreeBlock with no data (step S608-3), and receives an ACK packet indicative of "complete" from the printer (step S608-4). Thus, one write transaction ends.

Next, the printer returns the FreeBlockResponse. Similar to the command FreeBlock at step S608-1, the FreeBlockResponse is written as a response including the FreeBlockCount into the response register (step S608-5). An ACK packet indicative of "pending" is returned from the link layer of the image providing device (step S608-6). Then, the printer sends the FreeBlockResponse with no data (step S608-7), and receives an ACK packet indicative of "complete" (step S608-8). Thus, one write

transaction ends.

On the other hand, in the transfer method 2, only the FreeBlockResponse from the printer is used to obtain the FreeBlockCount at the second and the subsequent cycles in the print data block transfer. Accordingly, the FreeBlockCount is obtained only by the operation at steps S608-5 to S608-8.

The acquisition of the FreeBlockCount is necessary at each cycle of block transfer. Accordingly, in the transfer method 2, the number of packets transferred on the bus can be less than that in the transfer method 1.

Fig. 53 shows the commands WriteBlock in the transfer method 1 and the transfer method 2 in detail. As the command WriteBlock requires no response, commands in this procedure are sent in the order, the WriteBlock (step S609-1), an ACK packet indicative of "pending" (step S609-2), the WriteBlock with no data (step S609-3) and an ACK packet indicative of "complete" (step S609-4). The length of the procedure is the half of that in the case where commands and responses are transferred. This performs data transfer at a relatively high speed even if the processings in the link layer and the transaction layer are performed at a relatively slow speed.

Fig. 54 shows the command WriteBlock in the

transfer method 1 and the transfer method 2 in detail, in a case where the processings in the link layer and the transaction layer are performed at a sufficiently high speed. In this case, commands in this procedure are sent in the order, the WriteBlock (step S610-1), and an ACK packet indicative of "complete" (step S610-2). This performs more efficient data transfer.

Fig. 55 shows the flow of WriteBlock error processing upon occurrence of bus reset. In this case, the bus reset occurs at transfer of n-th (= 0-255) packet at one block transfer cycle. In a write transaction, a failure of data packet transfer is indicated by an ACK packet, however, error upon bus reset cannot be detected. Then, in the present embodiment, error processing is performed in the following procedure. That is, in a case where the FreeBlockCount is obtained (step S611-1), then the WriteBlock is sent n times (step S611-2 to S611-6), if bus reset occurs at this time (step S611-7), the image providing device again sends the WriteBlock[n] immediately before the bus reset (step S611-8), and thereafter, continues the processing (steps S611-9 to S611-14).

[Transfer Method 3]

Fig. 56 shows the construction of command registers, response registers and data registers of the image providing device and the printer, in the PUSH large buffer model.

The commands and responses between the image providing device and the printer based on the FCP protocol are executed by operation of writing a command frame 65-7, as command request data, from a command register 65-1 of the image providing device into a command register 65-4 of the printer, and operation of writing a response frame 65-8, as response data, from a response register 65-5 of the printer into a response register 65-2 of the image providing device.

Further, different from the FCP protocol, a data frame 65-9 is used in one-directional operation of writing image data from a data register 65-3 of the image providing device into a data register 65-6 of the printer by using a write transaction.

Fig. 57 shows the flow of operation of the PUSH buffer model between the image providing device and the printer.

Note that the commands Login, Logout, OpenChannel and CloseChannel and format setting are similar to those

in the above-described transfer method 1, therefore, detailed explanations of the commands and the format setting will be omitted.

In Fig. 57, the image providing device inquires about the buffer area of the printer by the command BufferConfig indicating "INQUIRY" (step S1701). The printer returns the buffer size and buffer address by the BufferConfigResponse (step S1702).

Next, the image providing device sets the buffer size and the buffer address of the printer into which data is written, by the command BufferConfig indicating "CONTROL" (step S1703). The printer returns the BufferConfigResponse indicating that the setting has been completed (step S1704).

Next, the image providing device notifies the printer that data transfer is to be started, by using the command SetBuffer indicating "NOTIFY" (step S1705). The printer returns the "INTERIM" SetBufferResponse indicating that the printer is prepared for the present to receive the data (step S1706), to let the image providing device start data transfer. Then, the printer notifies the image providing device that the data transfer to the initially set buffer area has been completed, by using the SetBufferResponse indicating

"CONTINUE" (step S1709).

The command WriteBuffer at step S1707 indicates data frame writing by the image providing device. In this operation, data is sequentially written into the buffer address set in the printer.

A response WriteTransactionResponse at step S1708 indicates a response packet upon isochronous transfer of data frame. As described above, if the data input speed of the printer is sufficiently high, the processing can be completed by using an acknowledgment of one write transaction, however, if data input takes time, independent responses occur as a split transaction.

Step S1710 indicates processing of transferring a plurality of data frames. That is, the data is transferred by a series of transactions to an area having the buffer size set by using the command BufferConfig. The data transfer method using a series of transactions is called a "PUSH data transfer method" or abbreviated to a "PUSH method".

Fig. 58 shows the structure of a data packet in the data frame. As the data can be written by directly addressing the buffer of the printer, a data packet 67-1 does not include a header and the like.

Fig. 59 shows the relation between the data

register and the buffer of the printer. Data written in a data register 68-1 is directly written into the address of a memory space 68-2 of the printer, designated by "BufferAddress" determined by an offset "Destination_Offset". As the offset value is incremented by a data length indicated by "Data_Length", the data is continuously written within the area indicated by "BufferSize" by repeatedly writing the data into the series of buffer addresses.

[Transfer Method 4]

Fig. 60 shows the construction of the command registers, the response registers and the data registers of the image providing device and the printer, in the PULL buffer model.

The commands and responses between the image providing device and the printer, based on the FCP protocol, are executed by operation of writing a command frame 69-7, as command request data, from a command register 69-1 of the image providing device into a command register 69-4 of the printer, and operation of writing a response frame 69-8, as response data, from a response register 69-5 of the printer into a response register 69-2 of the image providing device.

Further, different from the FCP protocol, a data frame 69-9 is used in one-directional operation of reading image data from a data register 69-3 of the image providing device into a data register 69-6 of the printer by using a read transaction.

Fig. 61 shows the flow of operation of the PULL buffer model between the image providing device and the printer. Note that the commands Login, Logout, OpenChannel and CloseChannel and format setting are similar to those in the above-described transfer method 1, therefore, detailed explanations of the commands and responses BufferConfig and SetBuffer (steps S1711 to S1714), which are the same as those in the above-described transfer method 3, will be omitted.

In Fig. 61, the image providing device notifies the printer that data transfer can be started by using the command SetBuffer indicating "NOTIFY" (step S1715). The printer returns the "INTERIM" SetBufferResponse indicating that the printer is prepared to receive data (step S1716) for the present, and let the image providing device start data transfer. Then, the printer notifies the image providing device that data transfer into the initially set buffer area has been completed, by using the SetBufferResponse indicating "CONTINUE"

(step S1719).

The printer requires a read transaction by a command PullBufferRequest (step S1717). Then the image providing device transfers data by a PullBufferResponse packet (step S1718), thus data is sequentially written into the buffer address set in the printer.

Step S1720 indicates processing of transferring a plurality of data frames. That is, the data is transferred by a series of read transactions to the area having the buffer size set by using the command BufferConfig. The data transfer method using a series of transactions is called a "PUSH data transfer method" or abbreviated to a "PUSH method".

Fig. 62 shows the relation between the data register and the buffer of the image providing device. Data is read from a memory space 72-2 of the image providing device designated by "BufferAddress" determined by an offset "Destination_Offset" set in the data register 71-1. As the offset value is incremented by a data length indicated by "Data_Length", the data is continuously read from the area indicated by "BufferSize" by repeatedly writing the data into the series of buffer addresses.

[Transfer Method 5]

In the transfer method 5 as the Isochronous model, the print data transfer by using the asynchronous transaction in the above-described transfer method 1 is replaced with print data transfer by using an isochronous transaction. Note that the structure of a data packet is the same as that shown in Figs. 46 and 47. Further, the data packet frame processing in the printer upon block transfer is the same as that in Fig. 48.

Note that according to the present transfer method, data transfer can be made at predetermined time by using an isochronous write transaction.

Further, in the block transfer, if an error occurs upon transferring print data for one page at once, it takes a long time to re-transfer the print data for one page. However, if print data is transferred in more fine block units, e.g., print-band units of an ink-jet printer, print data re-transfer due to the occurrence of error can be efficiently performed.

Fig. 63 shows command registers and response registers for flow control. The image providing device writes a command frame into a command register 75-3 of the printer by an asynchronous write transaction. The printer writes a response frame to the command into a

response register 75-2 of the image providing device by an asynchronous write transaction.

Fig. 64 shows the flow of print processing. First, similar to the above-described transfer method 1, the image providing device sends the command Login to the printer (step S507). The printer returns the LoginResponse (step S508), thus connection is established.

Next, similar to the above-described transfer method 1, the image providing device performs format setting (step S509), and sends the command OpenChannel to the printer (step S510). The printer returns the OpenChannelResponse (step S511), thus a logic channel is opened.

Then, the image providing device sends the command FreeBlock to the printer (step S512). The printer returns the FreeBlockResponse (step S513). The FreeBlockResponse includes the number of FreeBlock and ErrorStatus. The number of FreeBlock is the number of block buffers ensured in block units in the memory space of the printer. The ErrorStatus is used to notify the image providing device of error information in previous block transfer. Note that the printer always returns "normal" as the ErrorStatus to the first command

FreeBlock after the logic channel has been opened.

Then, the image providing device block-transfers print data by an isochronous transaction (step S514). At this time, the image providing device sends data packets of the number corresponding to the FreeBlockCount.

Next, the image providing device sends the command FreeBlock to the printer (step S515). The printer returns the FreeBlockResponse (step S516). If the ErrorStatus of the response indicates "abnormal", i.e., if an error has occurred at the previous block transfer, the image providing device sends the data transferred at step S514 again (step S517). Thereafter, the processing at steps S515 to S517 is repeated until the data transfer has been normally completed. Further, if the ErrorStatus indicates "normal", the image providing device sends data packets of the number indicated by the FreeBlockCount included in the FreeBlockResponse (step S517).

Note that the printer determines whether or not an error has occurred by referring to the block count of the header in transferred data (Fig. 64). Further, if the FreeBlockCount is greater than the number of blocks of data to be transferred, the image providing device sends dummy packets of a number corresponding to the

excessive blocks.

Then, data transfer by an isochronous transaction is repeated until all the series of print data have been outputted from the image providing device.

When the data transfer has been completed, similar to the above-described transfer method 1, the image providing device closes the logic channel by the command and response CloseChannel (steps S518a and S519), and logs out from the printer by the command and response Logout of the DPP protocol (steps S520 and S521).

As described above, according to the present embodiment, the image providing device and the printer are directly connected by using the 1394 serial bus or the like, and image data is directly sent from the image providing device to the printer, so that the printer prints an image based on the image data.

Further, as control commands and print data are separated, the embodiment provides an efficient data transfer methods in the 1394 serial bus or the like.

Further, the embodiment provides a data transfer method which recovers a transfer error in the 1394 serial bus.

Further, the embodiment provides a data transfer method in which determination whether or not data can be

written into the register area is unnecessitated by notification of the number of available blocks of a register area for data transfer, and the overhead necessary for the determination is removed. Further, the embodiment provide an efficient data transfer method since as data for the notified number of available blocks is transmitted and received.

Further, according to the embodiment, a data transfer method appropriate to a transfer destination device can be selected from a plurality of data transfer methods.

Further, the embodiment provides a data transfer method which avoids degradation of transfer efficiency due to command transmission upon data transfer from a host device to a target device, by only using a command instructing to start data transmission and a response to the command, i.e., transmitting no command after the start of the data transfer.

Further, the embodiment provides a data transfer method based on the PUSH method or PULL method upon data transfer between a host device and a target device.

Further, the embodiment provides a data transfer method which performs isochronous transfer and asynchronous transfer in the same transfer procedure

upon data transfer between a host device and a target device.

Further, the embodiment provides a data transfer method in which, when a transfer error occurs at a certain part of data in isochronous transfer, performs re-transfer the part of data where transfer the error has occurred, upon data transfer between a host device and a target device.

Further, the embodiment provides a data transfer method which performs proper data transfer even if bus reset occurs in data transfer between a host device and a target device.

Further, a peripheral device such as a printer which utilizes the above data transfer methods can be provided.

[Modification of Embodiment]

Note that the above embodiment has been described in a case where a network is constructed by using the IEEE 1394 serial bus, however, the present invention is not limited to the 1394 serial bus. For example, the present invention is applicable to a network constructed by using an arbitrary serial interface such as a Universal Serial Bus (USB).

In the above-described embodiment, commands and responses to the commands based on the FCP protocol are employed, and information is set at the response and notified to the host device. However, a method of mapping a register on a memory as the characteristic feature of the IEEE 1394 memory bus model can be considered.

In this case, a command is executed by writing command data into a command register assigned to a specific address of the memory. Similarly, a response is indicated by reading data at a response register assigned to a specific address of the memory.

Accordingly, when the target device recognizes that a command has written into a command register, the target device executes the command, and writes the result of the execution of the command and information into a response register. The host device which has written the command into the command register reads the response register of the target device and obtains the result of the execution of the command and information.

Thus, the present invention can be realized by using registers in the memory bus model.

In the above-described embodiment, discussing examples which have a target device as a printer.

However, the target device of the present invention is not limited to the printer. That is, some device recording image data such as a display apparatus and a storage apparatus applies to the target device of the present invention.

The present invention can be applied to a system constituted by a plurality of devices (e.g., host computer, interface, reader, printer) or to an apparatus comprising a single device (e.g., copy machine, facsimile).

Further, the object of the present invention can be also achieved by providing a storage medium storing program codes for performing the aforesaid processes to a system or an apparatus, reading the program codes with a computer (e.g., CPU, MPU) of the system or apparatus from the storage medium, then executing the program.

In this case, the program codes read from the storage medium realize the functions according to the embodiments, and the storage medium storing the program codes constitutes the invention.

Further, the storage medium, such as a floppy disk, a hard disk, an optical disk, a magneto-optical disk, CD-ROM, CD-R, a magnetic tape, a non-volatile type

memory card, and ROM can be used for providing the program codes.

Furthermore, besides aforesaid functions according to the above embodiments are realized by executing the program codes which are read by a computer, the present invention includes a case where an OS (operating system) or the like working on the computer performs a part or entire processes in accordance with designations of the program codes and realizes functions according to the above embodiments.

Furthermore, the present invention also includes a case where, after the program codes read from the storage medium are written in a function expansion card which is inserted into the computer or in a memory provided in a function expansion unit which is connected to the computer, CPU or the like contained in the function expansion card or unit performs a part or entire process in accordance with designations of the program codes and realizes functions of the above embodiments.

The present invention is not limited to the above embodiments and various changes and modifications can be made within the spirit and scope of the present invention. Therefore, to appraise the public of the

scope of the present invention, the following claims are made.